UNIVERSIDADE DO ALGARVE

FACULDADE DE CIÊNCIAS E TECNOLOGIA

Departamento de Engenharia Electrónica e Informática

Relatório da disciplina:

Periféricos e Interfaces

2º semestre 2005/2006

Helder Aniceto Amadeu de Sousa Daniel

Faro

Setembro de 2006

Índice

1 Introdução 1
2 Objectivos pedagógicos 1
3 Requisitos 1
4 Conteúdos programáticos 1
5 Programa 1
6 Bibliografia recomendada 2
6.1 Sobre device drivers e módulos para linux: 2
6.2 Sobre a camada de entrada saída do Sistema Operativo: 2
6.3 Sobre programação de baixo nível e BIOS: 2
6.4 Sobre hardware em geral: 2
6.5 Outras fontes de informação sobre programação de sistemas: 2
6.6 Resumos das aulas 3
7 Procedimento de avaliação 3
7.1 Épocas de avaliação 3
7.1.1 Época normal: 3
7.1.2 Época de recurso: 3
7.2 Admissão a exame 3
7.3 Aprovação na disciplina 3
8 Calendarização 3
8.1 Teórica 3
8.2 Prática 5
9 Assiduidade 6
9.1 Teóricas 6
9.2 Práticas 7
10 Avaliação 7
Enunciados dos momentos de avaliação 9

1 Introdução

Neste relatório encontra-se definido o conteúdo, objectivos, programa, bibliografia, métodos de avaliação da disciplina Periféricos e Interfaces, bem como uma análise da assiduidade e aprovação no ano lectivo de 2005-2006.
Esta disciplina consta do plano curricular dos cursos de Licenciatura em Informática e Licenciatura em Ensino de Informática, leccionados no Departamento de Engenharia Electrónica e Informática da Faculdade de Ciências e Tecnologia da Universidade do Algarve.
No plano curricular das licenciaturas acima referidas, esta disciplina insere-se no 2º semestre do 2º ano e correspondem-lhe 3 créditos.

Neste ano lectivo os objectivos pedagógicos mantiveram-se idênticos aos do ano lectivo anterior, foi no entanto alterado o programa para que a disciplina seja leccionada sobre o Sistema Operativo Linux.
Os capítulos interface IDE e SCSI, portos de comunicação e adaptador de monitor são versões actualizadas e adaptados ao Sistema Operativo Linux, a partir dos capítulos correspondentes leccionados nos dois anos lectivos anteriores, sobre o Sistema Operativo MS-DOS.
Todo omaterial lectivo respeitante ao ano de 2004-2005, desenvolvido pela Mestre Ana Paula Costa e pelo Professor Doutor José Bastos pode ser consultado em:

http://www.deei.fct.ualg.pt/PIn_2005/Anuncios/

2 Objectivos pedagógicos

Pretende-se dar a conhecer os princípios de funcionamento dos periféricos e interfaces padrão em computadores pessoais (PCs); dos dispositivos de armazenamento, como os discos magnéticos e ópticos; dos interfaces IDE e SCSI; dos periféricos de entrada e saídas de dados; dos barramentos ISA, PCI, USB.

3 Requisitos

Para frequentar esta disciplina recomenda-se ter já adquirido:

- Domínio dos conceitos da programação estruturada em ANSI C
- Domínio da implementação e utilização de estruturas de dados elementares
- Conhecimentos de programação em ASSEMBLER
- Conhecimentos de arquitectura de computadores, nomeadamente PCs
- Conhecimentos de organização e implementação de Sistemas Operativos, nomeadamente Linux.

4 Conteúdos programáticos

Drivers. Interfaces. Periféricos. Instalação e Configuração dos dispositivos referidos em PCs.

5 Programa

Abaixo apresenta-se o programa previsto para o semestre, conforme apresentado nas aulas teóricas:

1. Conceitos elementares: periférico, interface e device driver.
2. A camada de entrada saída de um sistema operativo.
3. Os principais componentes de um PC.
4. Microprocessadores Intel:

4.1. Descrição das arquitecturas 8086, 80286, 80386, 80486 e família Pentium.
4.2. Modos de funcionamento real, protegido e virtual 8086.
5. Chamadas de sistema (Linux system calls).
6. Princípios de Input / Output:
6.1. A memória principal.
6.2. Interrupções de hardware e DMA.
7. Módulos para kernel Linux.
8. Introdução à programação de device drivers em Linux:
8.1. Device drivers de bloco.
8.2. Device drivers de caracter.
9. CMOS, o relógio de tempo real, temporizadores.
10. Introdução aos barramentos PCI, ISA e USB.
11. Dispositivos de armazenamento (Discos).
12. O interface IDE e SCSI.
13. O porto paralelo.
14. O porto série.
15. O teclado e o rato.
16. O adaptador de monitor.

6 Bibliografia recomendada

6.1 Sobre device drivers e módulos para linux:

Corbet, Jonathan, Alessandro Rubini e Kroah-Hartman, Greg (2005*). "Linux Device Drivers 3rd edition"*, O'Reilly, http://lwn.net/Kernel/LDD3/

Salzman, Peter Jay, Michael Burian and Ori Pomerantz (2005). *"The Linux Kernel Module Programming Guide"*, *http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf*

6.2 Sobre a camada de entrada saída do Sistema Operativo:

Tanenbaum, Andrew S. (2001). *"Modern Operating Systems. 2nd Edition"*, Prentice Hall

Silberschatz, Avi, Peter Baer Galvin and Greg Gagne (2004). *"Operating System Concepts. 7th Edition"*, John Wiley & Sons

6.3 Sobre programação de baixo nível e BIOS:

Gilluwe, Frank van (1997). *"The Undocumented PC - A Programmer's Guide to I/O, CPU's and Fixed Memory Areas 2nd edition*", Addison-Wesley

Johnson, Marcus (1994). *"PC Programmer's Guide to Low-Level Functions and Interrupts"*, Sams publishing

6.4 Sobre hardware em geral:

Messmer, Hans-Peter (2002). *"The Indispensable PC Hardware Book 4th edition"*, Addison-Wesley

6.5 Outras fontes de informação sobre programação de sistemas:

Tanenbaum, Andrew S. e Albert S. WoodHull (2006). *"Operating Systems Design and Implementation 3rd edition"*, Prentice-Hall.

Tanenbau, Andrew S. (2006). *"Structured Computer Organization, 5th Edition"*, Prentice Hall.

Karbo, Michael (2005). *"PC Architecture"*, http://www.karbosguide.com

Tischer, Michael e Bruno Jennrich, (1996). *"PC Intern - The Encyclopedia of System Programming 6th edition",* Abacus

Wilton, Richard (1994). *"Programmer's Guide to PC Video Systems"*, Microsoft Press

Norton, Peter, Peter Aitken e Richard Wilton, (1993), *"PC Programmer´s Bible*", Microsoft Press

6.6 Resumos das aulas

Resumos das aulas, assim como outros elementos relativos à disciplina, estão disponíveis *on-line* em http:/w3.ualg.pt/~hdaniel/pin

Resumos das aulas ano lectivo 2004-2005 http://www.deei.fct.ualg.pt/PIn_2005/Anuncios/
Resumos das aulas ano lectivo 2003-2004 http://w3.ualg.pt/~apcosta/pin/index.html

7 Procedimento de avaliação

7.1 Épocas de avaliação

A avaliação é efectuada em duas épocas:

7.1.1 Época normal:

A avaliação na época normal é a média ponderada do exame de época normal e dos trabalhos práticos:

Avaliação época normal = 70% Exame época normal + 30 % Trabalhos práticos

7.1.2 Época de recurso:

Na avaliação da época de recurso o aluno pode optar unicamente pela avaliação no exame de recurso, ou tal como na época normal a média ponderada do exame de época de recurso e dos trabalhos práticos.

7.2 Admissão a exame

Todos os alunos regularmente inscritos são admitidos aos exames.
Os exames consistem de prova escrita, sem consulta, podendo incluir questões sobre conhecimentos necessários à realização do trabalho prático.

7.3 Aprovação na disciplina

Serão aprovados na disciplina alunos com nota igual ou superior a 10 valores, em qualquer das épocas.

8 Calendarização

A disciplina é composta por uma componente teórica semanal de 2 horas e uma componente prática semanal de 3 horas. A componente teórica divide-se em dois períodos não consecutivos de 1 hora.

8.1 Teórica

A exposição da matéria teórica, segundo o programa apresentado no ponto 5, foi calendarizada para 29 aulas teóricas, com duração aproximada de 50 minutos:

1. Apresentação.
2. Conceitos elementares. Componentes do PC. Barramentos.
3. A Unidade de processamento central: Intel 8086/88.
4. A Unidade de processamento central: Evolução dos CPU usados em PCs.

5. Chamadas ao sistema operativo Linux.
6. Modelo de Entrada / Saída I.
7. Modelo de Entrada / Saída II.
8. Introdução ao desenvolvimento de módulos do núcleo Linux (LKMs I).
9. Introdução ao desenvolvimento de módulos do núcleo Linux (LKMs II).
10. Introdução aos Drivers para dispositivos de caracter: registo do número de dispositivo.
11. Introdução aos Drivers para dispositivos de caracter: registo do driver no kernel, funções open e release.
12. Introdução aos Drivers para dispositivos de caracter: funções read e write I.
13. Introdução aos Drivers para dispositivos de caracter: funções read e write II, alocação dinâmica de memória no kernel.
14. Exemplo de driver para RTC.
15. Acesso à configuração do PC na memória CMOS.
16. IOCTL I.
17. IOCTL II.
18. Drivers para dispositivos de bloco: Registo do dispositivo.
19. Drivers para dispositivos de bloco: Exemplo de driver para ramdisk.
20. Drivers para dispositivos de bloco: Exemplo de driver para ramdisk com simulação de suporte (disco) removível.
    Formatação de discos magnéticos: propriedades estáticas.
21. Formatação de discos magnéticos: propriedades dinâmicas. Particionamento de discos rígidos.Arrays de discos (RAID).
22. Discos ópticos.
23. Interface IDE e SCSI.
24. Floppy Disk controller.
    Porto Série I.
25. Porto Série II: a UART. Porto paralelo.
26. Sub-sistema de vídeo: introdução, modos gráficos, memória de video.
27. Sub-sistema de vídeo: registos da SVGA.
28. Sub-sistema de vídeo: bitplanes nos vários modos gráficos.
29. Sub-sistema de vídeo: acesso directo à memória de video. Introdução à SVGALIB

PIn 2005-6 **Teóricas**

			T1	
Semana	**Inicio**	**Fim**	**QUI**	**SEX**
1	13-Fev-05	17-Fev-05	1	**FLT**
2	20-Fev-05	24-Fev-05	2	3
3	27-Fev-05	03-Mar-05	4	5
4	06-Mar-05	10-Mar-05	6	7
5	13-Mar-05	17-Mar-05	8	9
6	20-Mar-05	24-Mar-05	10	11
7	27-Mar-05	31-Mar-05	12	13
8	03-Abr-05	07-Abr-05	14	15
	10-Abr-05	14-Abr-05	**Páscoa**	
9	17-Abr-05	21-Abr-05	16	17
10	24-Abr-05	28-Abr-05	18	19
11	01-Mai-05	05-Mai-05	20	21
	08-Mai-05	12-Mai-05	**Semana académica**	
12	15-Mai-05	19-Mai-05	22	23
13	22-Mai-05	26-Mai-05	24	25
14	29-Mai-05	02-Jun-05	26	27
15	05-Jun-05	09-Jun-05	28	29

Os alunos dos cursos aos quais a disciplina foi leccionada foram agrupados num turno teórico, referido por T1, leccionado em dois períodos de uma hora por semana, sendo atribuído o serviço docente do seguinte modo:

T1 Helder Daniel (professor auxiliar DEEI)

As aulas foram leccionadas segundo a calendarização acima apresentada.

A aula teórica prevista para o turno T1 no dia 17 de Fevereiro de 2006, não foi leccionada por ausência do docente, por necessidade de apoio à família, no entanto o programa previsto foi cumprido.

8.2 Prática

Nas aulas práticas, além do tempo dedicado aos trabalhos práticos, foram resolvidos exercícios e implementadas pequenas aplicações sobre os temas apresentados nas aulas teóricas, distribuídos por 7 fichas:

1. Identificação dos componentes de um PC
2. Introdução ao ambiente de trabalho
3. Chamadas ao sistema
4. Acesso aos portos de E/S. Introdução aos módulos do núcleo Linux.
5. Drivers para dispositivos de caracter
6. Drivers para dispositivos de bloco
7. Sub-sistema de video

PIn 2005-6 **Práticas**

Semana	Inicio	Fim	Seg P1	Ter P4	Qua P2	Qui P3
1	13-Fev-05	17-Fev-05	Não se efectuam aulas práticas			
2	20-Fev-05	24-Fev-05	1	1	1	1
3	27-Fev-05	03-Mar-05	Carnaval			2
4	06-Mar-05	10-Mar-05	2	2	2	3
5	13-Mar-05	17-Mar-05	2/3	FLT	2/3	Semana Aberta
6	20-Mar-05	24-Mar-05	3	2/3	3	3/4
7	27-Mar-05	31-Mar-05	4	3	4	4
8	03-Abr-05	07-Abr-05	5	-	Apoio	5
	10-Abr-05	14-Abr-05	Páscoa			
9	17-Abr-05	21-Abr-05	Páscoa	-	Apoio	5
10	24-Abr-05	28-Abr-05	Apoio	Fer	Apoio	5
11	01-Mai-05	05-Mai-05	Fer	-	Apoio	5/TP1
	08-Mai-05	12-Mai-05	Semana académica			
12	15-Mai-05	19-Mai-05	Apoio	-	Apoio	6
13	22-Mai-05	26-Mai-05	Apoio	-	Apoio	6
14	29-Mai-05	02-Jun-05	Apoio	-	Apoio	6/TP2
15	05-Jun-05	09-Jun-05	Apoio	-	Apoio	7

Os alunos dos cursos aos quais a cadeira é leccionada foram divididos em 4 turnos, P1 – P4, sendo distribuído o serviço docente do seguinte modo:

P1 Daniel Castro (assistente convidado DEEI)
P2 Daniel Castro (assistente convidado DEEI)
P3 Helder Daniel (professor auxiliar DEEI)
P4 Amine Berqia (professor auxiliar DEEI)

As fichas foram resolvidas nas aulas conforme mostra a calendarização acima. Na primeira semana não foram efectuadas aulas práticas por não ter sido leccionada matéria teórica.
O turno P4 fechou partir da 8ª semana devido a ausência de alunos.
Os turnos P1 e P2 deixaram de ter ausência regular dos alunos nele inscritos a partir da 10ª e da 8ª semana respectivamente, tendo servido de aulas de apoio às fichas e trabalhos práticos a qualquer aluno inscrito na disciplina.
Nas aulas práticas da 10ª e 14ª semanas do turno P3, foi dado apoio à conclusão do 1º e 2º trabalhos práticos respectivamente.

A avaliação prática foi dividida por dois trabalhos práticos. Ambos consistiram na implementação de módulos para o sistema operativo GNU/Linux sobre o kernel 2.6, na linguagem C.
O primeiro, referido por TP1, consistiu na implementação de um driver elementar para manipulação do relógio de tempo real.
O segundo, referido por TP2, num módulo para leitura de informação de um dispositivo de bloco: um disco rígido IDE.

9 Assiduidade

9.1 Teóricas

Alunos inscritos na disciplina segundo os serviços académicos:

Curso	
ESC	1
EI	35
I	57
Total	**93**

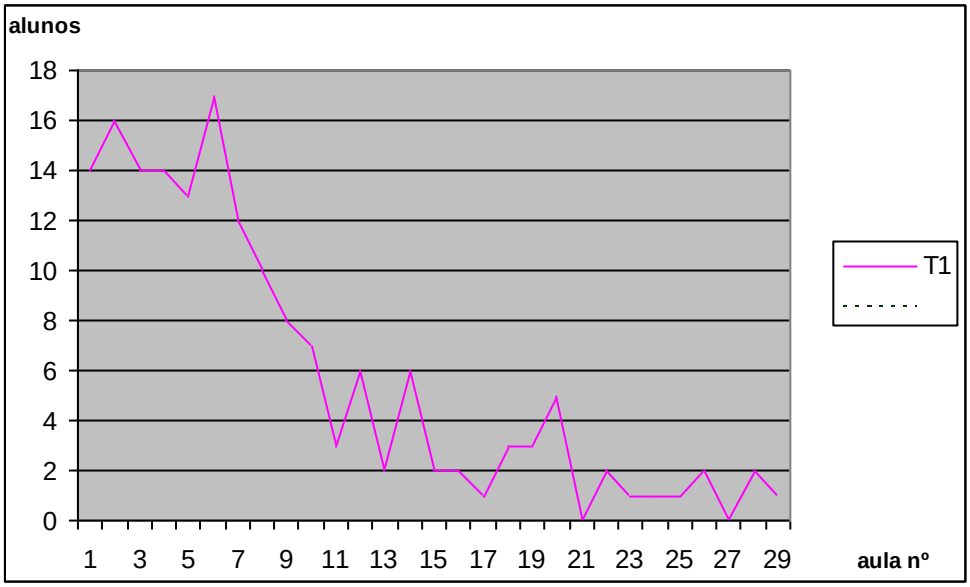


Fig. 1 – Assiduidade às aulas teóricas

No gráfico acima têm-se a assiduidade à componente teórica.

9.2 Práticas

Alunos inscritos nos turnos práticos:

Turno	
P1	7
P2	9
P3	5
P4	15
Total	36

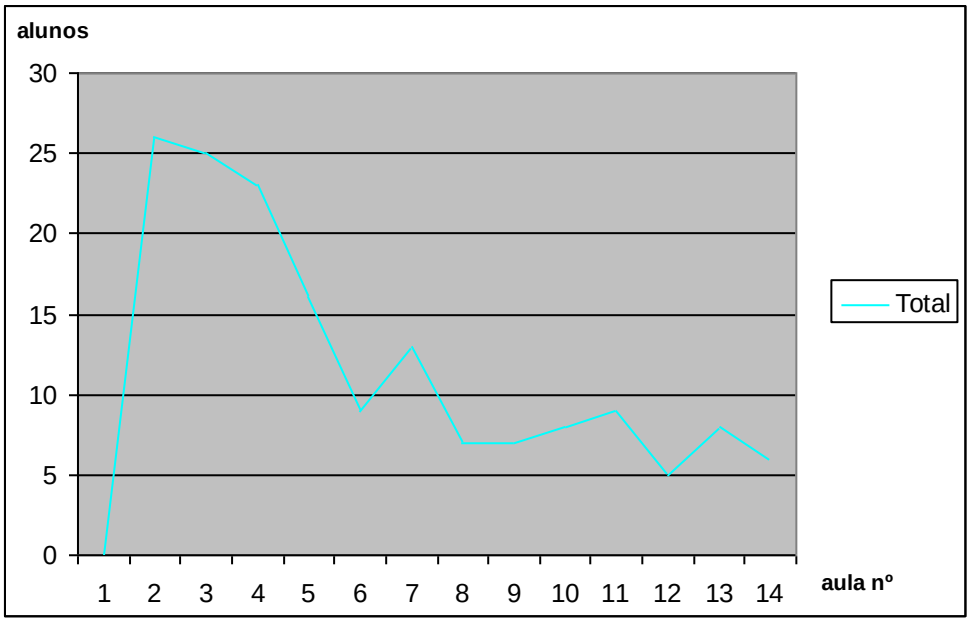


Fig. 2 – Assiduidade às aulas práticas

No gráfico acima têm-se o sumatório da assiduidade aos quatro turnos práticos por aula.

10 Avaliação

Para avaliação contínua, exame de época normal e exame de recurso está indicada a percentagem de aprovação baseada nos alunos que se propuseram a cada época de avaliação, e não a totalidade de inscritos segundo os serviços académicos.

	ESC	EI	I	**Total**
Entregaram o 1º Trabalho:	0	0	9	**9**
Entregaram o 2º Trabalho:	0	0	7	**7**
Assiduidade ao exame época normal:	0	1	10	**11**
Aprovados no exame época normal:	0	0	5	**5**
% aprovação:	0,00%	0,00%	50,00%	**45,45%**
Assiduidade ao exame recurso:	0	0	5	**5**
Aprovados no exame recurso:	0	0	3	**3**
% aprovação:	0,00%	0,00%	60,00%	**60,00%**
Total de alunos aprovados				**8**

Os seguintes histogramas mostram a distribuição de notas para as épocas de avaliação acima indicadas:

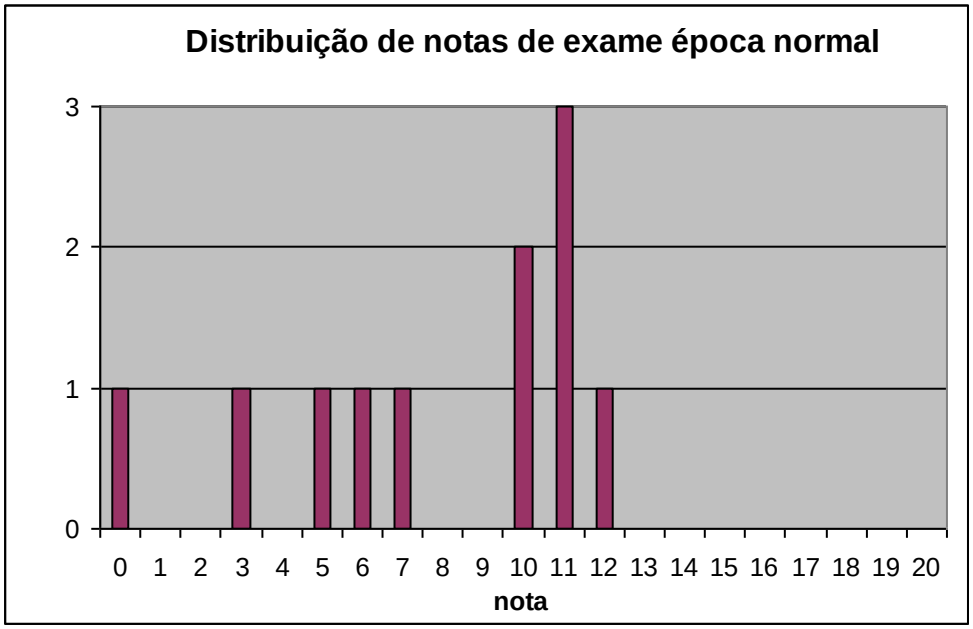


Fig. 3 – Distribuição de notas de época normal

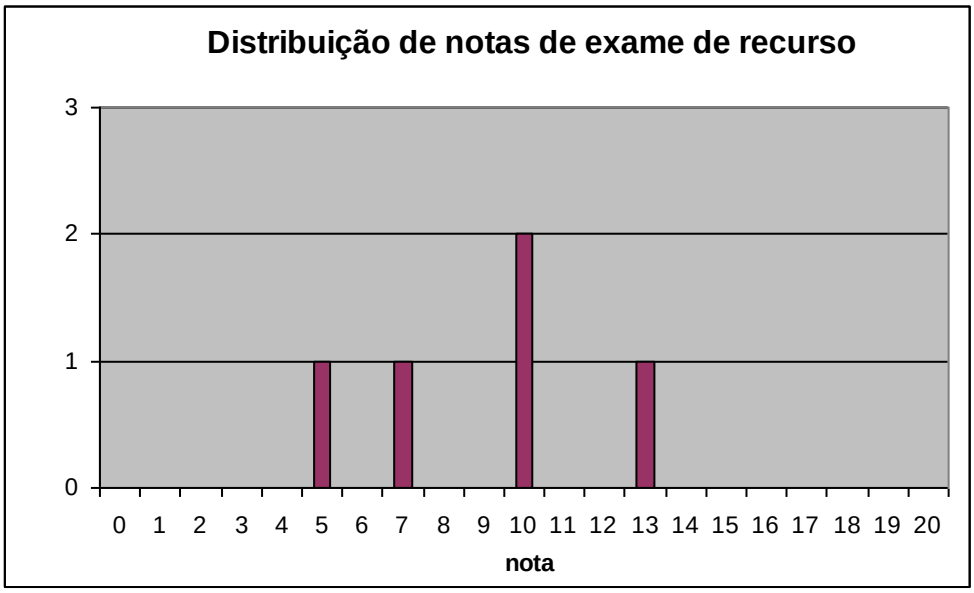


Fig. 3 – Distribuição de notas de época de recurso

Helder Daniel
Responsável pela disciplina de Periféricos e Interfaces no ano lectivo 2005/2006

Enunciados dos momentos de avaliação

Exame época normal
Exame recurso
1º Trabalho prático
2º Trabalho prático

Universidade do Algarve
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrónica e Informática

Periféricos e Interfaces
2005/2006
Curso de Informática e Ensino de Informática

Exame época normal 28 de Junho de 2006 **Duração**: 2 horas

Sem consulta. É fornecida informação técnica adicional em anexo.

1. (1v) Distinga dispositivo de caracter de dispositivo de bloco. Dê um exemplo de cada um.

2. A função da libc: **ssize_t write(int fd, const void *buf, size_t count)** é um wrapper para uma chamada ao sistema. Escreva uma possível implementação desta função, não esquecendo o valor retornado, em:

   a) (1v) C, usando a função da libc **syscall**.
   b) (1v) Assembler embebido em C

3. (1.5v) Explique o porquê do baixo desempenho das transferências de dados quando executadas segundo o principio de entrada saída programada.

4. (1.5v) Indique outra estratégia de transferência de dados com maior desempenho que a referida na questão anterior, explicando o porquê.

5. (2v) Pretende-se ler a hora corrente do RTC, no espaço do utilizador. Escreva um segmento de código que acedendo aos registos do RTC/CMOS mostre a hora no formato HH:MM:SS.

6. (1.5v) Escreva um segmento de código de um módulo, que permita ler o valor do parâmetro **count** da linha de comandos:

```
insmod module.ko count=512
```

E escrevê-lo no registo do sistema.

7. (1.5v) Explique porque o tempo médio de acesso numa drive de CD é maior do que num disco rígido.

8. (1.5v) Explique sucintamente o que entende por interleaving e a sua utilidade.

9. (1.5v) Considerando um disco rígido já formatado, quais as diferenças entre o Master Boot record e um Boot Sector.

10.
a) (2.5v) Explique detalhadamente o segmento de código seguinte:

```
// …
#define outp(port,byte) outb(byte,port)

// registos do Sequenciador
outp (0x3C4, 0x00); // reset do
outp (0x3C5, 0x01); // Sequenciador

outp (0x3C4, 0x02); // Map Mask
outp (0x3C5, 0x04);

outp (0x3C4, 0x04); // Memory Mode
outp (0x3C5, 0x07);

outp (0x3C4, 0x00); // Set do
outp (0x3C5, 0x03); // Sequenciador

// registos do controlador gráfico
outp (0x3CE, 0x04); //  Read Map Select
outp (0x3CF, 0x02);

outp (0x3CE, 0x05); //  Graphics Mode
outp (0x3CF, 0x00);

outp (0x3CE, 0x06); //  Miscellaneous
outp (0x3CF, 0x00);
// …
```

b) (1v) Se posteriormente for executado o seguinte código o que sucederá:

```
// …
      unsigned char *vaddr;

      f=open ("/dev/mem", O_RDWR);
      if (f<0) { printf("File not found\n"); exit(1); }
      vaddr=(unsigned char*)mmap(0, 8192, PROT_READ | PROT_WRITE,
                                          MAP_SHARED, f, 0xA0000);
      vaddr['A'*32]=255;
// …
```

11. (2.5v) Escreva um segmento de código que se execute no espaço de utilizador e desloque o caracter ‘*’ (com cor branca e fundo vermelho) sobre as 5 primeiras linhas do monitor em modo de texto 80x25, sem alterar o conteúdo prévio dessas linhas.

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp1.pdf

1º Trabalho Prático PIn 2005/06
Driver elementar para RTC

1 Descrição

Desenvolver um driver de caracter para o relógio de tempo real (RTC), que permita consultar e alterar a hora e data de acordo com as especificações seguintes:

1.1

O driver deverá chamar-se **rtci**, de modo que deverá ser identificado em **/proc/devices** por:

```
<Major>     rtci
```

1.2

Terá um parâmetro de linha de comando que permita que o utilizador indique o número Major do dispositivo:

```
insmod rtci.ko [Major = ??]
```

No caso do parâmetro ser omitido o Major deverá ser automaticamente atribuído.
No caso do Major especificado pelo utilizador não estar livre o driver não deverá ser inserido, sendo enviada uma mensagem para o registo do sistema com **printk( )**.

1.3

Quando o driver for inserido ou removido deverá também ser registada a operação com **printk ()**.

1.4

Deverá ser escrito um script, chamado **rtci**, que receba como primeiro parâmetro **start** ou **stop**, e que respectivamente insira ou remova o driver no kernel. Como segundo parâmetro (opcional), este script poderá receber o Major do driver especificado pelo utilizador:

```
rtci  <start | stop>  [Major = ??]
```

Que deverá ser passado para o driver no momento da inserção.

1.5

Os serviços disponibilizados pelo driver deverão ser:

a) **open** e **release**, de modo que garantam que o driver não poderá ser acedido por mais que um utilizador ao mesmo tempo.

b) **read** que indica a data e hora actual do sistema, no formato:

```
cat /dev/rtci

<dia da semana>   DD/MM/AAAA-HH:MM:SS  <AM | PM>
```

Por exemplo, para o modo horário de 24 horas:

```
Terça-feira 18/04/2006-20:02:12
```

e para o modo horário de 12 horas:

```
Terça-feira 18/04/2006-08:02:12 PM
```

c) **write** que permite alterar a data e hora do sistema, no formato:

```
cat > /dev/rtci

DD/MM/AAAA-HH:MM:SS
^D
```

A string acima assume sempre o modo horário de 24 horas. Se o modo corrente do RTC for 12h deverá ser tomado isso em consideração de modo que a hora seja correctamente ajustada.
Uma opção será passar para o modo 24 horas, ajustar a hora e voltar ao modo inicial.
Outra poderá ser subtrair 12 à hora, se esta for maior que 12 horas, e seleccionar o bit mais alto do byte da hora do RTC.

A opção utilizada, podendo ser outra além das duas acima indicadas desde que funcional, deverá ser justificada no relatório.

d) **ioctl** que permita alterar o modo horário de 24 horas para 12 horas e que permita também invocar as operações de escrita e leitura. O ficheiro header a utilizar, com nome **"rtcioctl.h"** deverá ser:

```
/* IOCTLs para rtci
*/

#ifndef RTCIOCTL
#define RTCIOCTL

#include <linux/ioctl.h>

#define IOCID 0x16

// argumento=1 - set 24h, 0 - set 12h
#define RTC_SET_24H  _IOW   (IOCID, 1, int)

// return 1 se modo 24h ou 0 se 12h
#define RTC_GET_24H  _IO    (IOCID, 2)

// Permite ajustar a hora, com a string passada como argumento
// Retorna o nº de bytes transferidos ou <0 em caso de erro
#define RTC_SET      _IOR   (IOCID, 3, char *)

// Retorna na string argumento a data e hora
// Retorna o nº de bytes transferidos ou <0 em caso de erro
#define RTC_GET      _IOW   (IOCID, 4, char *)

#endif
```

1.6
Deverá ser desenvolvido também um programa em C, que possa ser executado por qualquer utilizador, e que aceda ao RTC através do driver. O programa deverá chamar-se **rtcm.c** e suportar os seguintes parâmetros na linha de comandos:

```
rtcm 12h                    Selecciona modo de 12 horas
rtcm 24h                    Selecciona modo de 24 horas
```

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp1.pdf

```
rtcm read                       Imprime data e hora no formato
                                indicado em 1.5 b)

rtcm write DD:MM:AA-HH:MM:SS    Ajusta data e hora no formato
                                indicado  1.5 c)
```

As operações de alteração do modo horário e de leitura e escrita, deverão ser efectuadas usando IOCTL, de acordo com os comandos especificados em 1.5 d).
Se o número de parâmetros não for o correcto, ou não for conhecido deverá ser impressa a mensagem:

```
Usar com:
rtcm [read | 24h | 12h ]
rtcm write DD/MM/AAAA-HH:MM:SS
```

2 Entrega do trabalho

2.1 Data

A data limite de entrega é terça-feira 16 de Maio de 2006. A entrega para além do prazo sofre uma penalização de 0.5 valores por dia.

2.2 Código fonte

O ficheiro com o código fonte do driver deverá ter o nome **rtci.c** e deve ser acompanhado de um *makefile* para gerar o módulo com o nome **rtci.ko**. O módulo deverá ser compilado de modo que com o comando **modinfo** obtenha-se informação pelo menos sobre a licença, a descrição dos parâmetros (indicando o valor por defeito), uma descrição breve do driver, a versão, e os Nomes, N$^{os.}$ e curso dos autores:

```
modinfo rtci.ko

filename:       rtci.ko
license:        GPL
author:         Helder Daniel
version:        1.0
description:    Simple RTC driver
vermagic:       2.6.14-ipipe preempt 386 gcc-3.3
depends:
srcversion:     B805AF6B7B4FFDBB3C1CE8E
```

De qualquer forma, todos os ficheiros devem ter como comentário os Nomes, N$^{os.}$ e curso dos elementos do grupo.
Deve ser entregue também o script **rtc.i** que insere e remove o módulo do kernel, referido em 1.4.

Pode ser entregue por e-mail ou em disquete ao docente do turno prático em que está inscrito o grupo.

2.3 Relatório

Com o código fonte terá de ser entregue um relatório, que deverá incluir um pequeno manual do utilizador e um manual de implementação. Um *template* para este relatório está disponível na página da disciplina:

http://w3.ualg.pt/~hdaniel/pin/avaliacao/relat_template.pdf

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp1.pdf

No manual do utilizador deverá ser descrito tanto o driver **rtci.ko** como a aplicação **rtcm**, referida no ponto 1.6, incluíndo os parâmetros da linha de comandos de ambos.

O relatório deverá ser entregue em papel, ao docente do turno prático em que está inscrito o grupo.

3 Avaliação

De acordo com o indicado na aula teórica 1. Será tomada especial atenção à correcção do programa, à eficiência da implementação e à estrutura e clareza do código.

Nota: Funcionalidades não pedidas nos enunciados não serão avaliadas.

4 Bibliografia recomendada

Corbet, Jonathan, Alessandro Rubini e Kroah-Hartman, Greg (2005). *"Linux Device Drivers 3rd edition"*, O'Reilly, http://lwn.net/Kernel/LDD3/

Salzman, Peter Jay, Michael Burian and Ori Pomerantz (2005). *"The Linux Kernel Module Programming Guide"*, *http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf*

Aulas teóricas e práticas disponíveis em http://w3.ualg.pt/~hdaniel/Ed/, especialmente:

Capítulo 5 e 6 das aulas teóricas.
Ficha prática 4.2 e 5.

Nota:
podem ser utilizados ou adaptados funções e scripts desenvolvidas nas aulas teóricas, práticas ou dadas como código auxiliar para as fichas práticas.

Universidade do Algarve
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrónica e Informática

Periféricos e Interfaces
2005/2006
Cursos: Informática, Ensino de Informática, Engenharia de Sistemas e Informática

Exame época recurso 18 de Julho de 2006 **Duração**: 2 horas

Sem consulta. É fornecida informação técnica adicional em anexo.

Nota: Ao longo deste exame considere-se o kernel Linux 2.6.xx e código C (gcc 3.3.5 ou superior e a biblioteca libc6)

1. (1v) Uma impressora, conectada a um PC através do porto paralelo classifica-se como um dispositivo de bloco ou de caracter? Justifique.

2. (1v) Represente esquematicamente como é efectuada a comunicação entre o CPU e a RAM e o CPU e os periféricos ISA e PCI, para arquitecturas de processadores de 32 bits, não esquecendo o hardware necessário para encaminhar a informação para os diferentes baramentos.

3. A função da libc: **int open (const char *fname, int flags, mode_t mode)** é um wrapper para uma chamada ao sistema. Escreva uma possível implementação desta função, não esquecendo o valor retornado, em:

a) (1v) C, usando a função da libc **syscall**.
b) (1v) Assembler embebido em C

4. (1.5v) Distinga entrada-saída por DMA e por interrupção.

5.

a) (3.5v) Implementar a função **int rtc_ioctl (struct inode *node, struct file *filp, unsigned int cmd, unsigned long par)**, de um driver para RTC/CMOS para kernel Linux, que permita:

   i) Se cmd == 0x100 e o modo horário corrente for 12h passar para 24h. Se o modo horário corrente for 24h passar para 12h.
   ii) Se cmd == 0x101 retorna 1 se o modo horário corrente for 12h ou 0 se for 24h.
   iii) Se cmd == 0x102 retorna 1 se a bateria dos CMOS está carregada ou 0 caso contrário

b) (1v) Os comandos suportados na alínea anterior foram definidos com os inteiros 0x100-0x102. Se bem que funcional esta forma de atribuir códigos aos comando não é a mais segura. Explique porquê.

6. (1.5v) Descreva o processo de procura e leitura de um sector num disco rígido, referindo como se posicionam as cabeças de leitura escrita para ler um sector especificado por uma localização CHS (Cylinder, Head, Sector).

7. (1.5v) Explique porque as unidades de disco rígido actuais têm um factor de interleaving 1:1.

8.

a) (1.5 v) Escreva código no espaço de utilizador que indique se as unidades de disco flexível 0 e 1 estão presentes ou não no sistema, usando a informação guardada na memória CMOS.

b) (1v) Altere o código anterior para se alguma das unidades estiver presente no sistema indicar o seu tipo (720KB, 1.44MB, etc...)

9.

a) (2.5v) Assumindo que se está num modo gráfico com 200 linhas, 320 colunas, 256 cores e que p aponta o inicio da memória de vídeo, explique detalhadamente o segmento de código seguinte:

```
// …
#define cols 320
unsigned char block[64];

void move(int l, int c, unsigned char* p) {
unsigned char tmp;
int i, j, vidx, fidx;

for (j=0;j<8; j++)
      for (i=0;i<8; i++) {
            vidx=(l+j)*cols+c+i;
            fidx = j*8+i;
            if (block[fidx]!=255) {
                  tmp=p[vidx];
                  p[vidx]=block[fidx];
                  block[fidx]=tmp; }
      }
};

int main() {
   unsigned char *p; //virtual address of video RAM
   int i;
// …
   for (i=0; i<64; i++) block[i]=4;            //4 == laranja
   for (i=24+2; i<24+6; i++) block[i]=255;     //255 == preto
   for (i=32+2; i<32+6; i++) block[i]=255;

   for (i=0;i<1000;i++) p[rand()%10000]=rand()%16;
   for (i=10;i<100;i++) { move(10,i,p); usleep(1000); move(10,i, p); }
// …
```

b)(1v) Explique o que sucederia se na função **move (...)** a linha de código:

```
      if (block[fidx]!=255) {
```

fosse substituída por: `if (block[fidx]!=4) {`

10. (1v) Considerando modo texto de 16 cores, descreva como é guardada no buffer de vídeo a informação sobre cada caracter que aparece no ecran.

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp2.pdf

2º Trabalho Prático PIn 2005/06
Módulo do kernel para leitura de disco IDE

1 Descrição

Desenvolver um módulo do kernel, segundo a metodologia dos drivers para dispositivos de caracter, que permita ler informação do disco rígido master ligado ao controlador IDE primário, de acordo com as especificações seguintes:

1.1

O módulo deverá chamar-se **dsk**, de modo que deverá ser identificado em **/proc/devices** por:

```
Block devices
(...)
<Major>     dsk
```

1.2

Terá um parâmetro de linha de comando que permita que o utilizador indique o número Major do dispositivo:

```
insmod dsk.ko [Major = ??]
```

No caso do parâmetro ser omitido o Major deverá ser automaticamente atribuído.
No caso do Major especificado pelo utilizador não estar livre o módulo não deverá ser inserido, sendo enviada uma mensagem para o registo do sistema com **printk( )**.

1.3

Quando o módulo for inserido ou removido deverá também ser registada a operação com **printk ()**.

1.4

Deverá ser escrito um script, chamado **dsk**, que receba como primeiro parâmetro **start** ou **stop**, e que respectivamente insira ou remova o módulo no kernel. Como segundo parâmetro (opcional), este script poderá receber o Major do módulo especificado pelo utilizador:

```
dsk  <start | stop>  [Major = ??]
```

Que deverá ser passado para o módulo no momento da inserção.

1.5

Os serviços disponibilizados pelo módulo deverão ser:

a) **open** e **release**, de modo que actualizem um contador com o número de utilizadores correntes.

b) Leitura de sectores. Assim com o comando:

```
hexedit –s /dsk/
```

deve ser possível visualizar os sectores.

A escrita de sectores, no entanto, deve ser impedida. Se se tentar escrever, p.ex. com **hexedit,** ou com **cat /dev/dsk**, deve ser escrita a mensagem no registo do **kernel:**

```
DSK device is Read only
```

c) **ioctl** que permita saber o número de utilizadores correntes, a versão ATA/ATAPI do dispositivo, o número de sectores total do disco (em modo LBA) e o tamnho em bytes de cada bloco ou sector. O ficheiro header a utilizar, com nome **"dskioctl.h"** deverá ser:

```
/* IOCTLs para dsk
*/

#ifndef DSKIOCTL
#define DSKIOCTL

#include <linux/ioctl.h>

#define IOCID 0x17

// return no. users
#define HDIO_NUSERS       _IO   (IOCID, 0)

// return ATA/ATAPI version
#define HDIO_ATAVER       _IO   (IOCID, 1)

// return no. LBA sectors in drive
#define HDIO_LBASEC   _IO   (IOCID, 2)

// return sectors (block) size
#define HDIO_SECSIZE  _IO   (IOCID, 3)

#endif
```

Sugestão: Identificar o disco IDE0 na inicialização do módulo, guardando os valores necessários para a versão ATA/ATAPI e nº de sectores no disco em variáveis, que serão retornadas pelos comandos de IOCTL. Assim evita-se efectuar a identificação do disco sempre que é necessário executar um comando IOCTLm como HDIO_ATAVER e HDIO_LBASEC.

1.6
O módulo não deverá suportar remoção de disco.

1.7
Deverá ser desenvolvido também um programa em C, que possa ser executado por qualquer utilizador, e que aceda ao módulo. O programa deverá chamar-se **dskm.c** e suportar os seguintes parâmetros na linha de comandos:

```
users       Retorna o número de utilizadores correntes do módulo dsk:

                  Users: 1

sectors     Retorna o número de sectores total do disco IDE0

                  LBA maximum sectors: 32905939205

secsize     Retorna o número de bytes em cada sector

                  Sector (block) size: 512
```

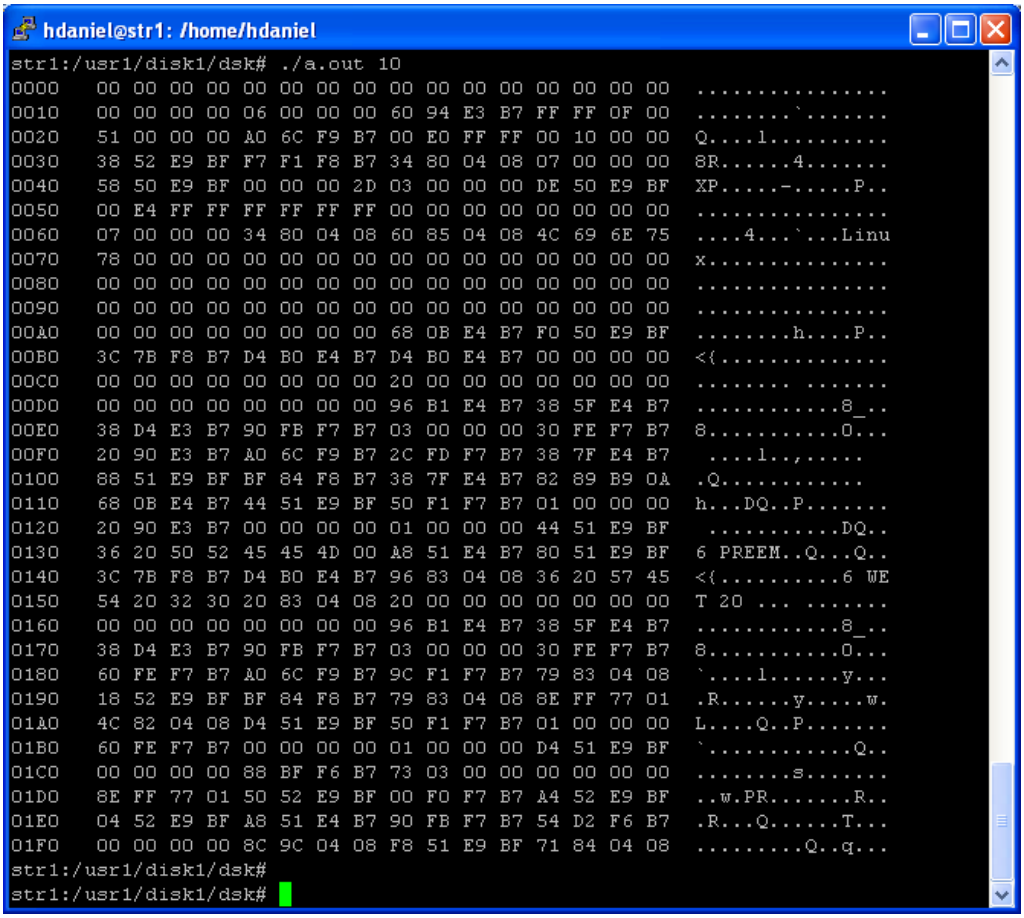
`version` Retorna a versão ATA/ATAPI

ATA/ATAPI version: 6

`geo` Retorna a geometria CHS lógica do disco IDE0, onde se assumem 255 cabeças e 63 sectores por cabeça

C: 2048 H: 255 S: 63

`<nº sector>` Apresenta o sector passado como argumento na forma:



Isto é apresenta o valor de cada byte em hexdecimal e em ASCII. Se o código ASCII for menro que 32 ou superior a 127 mostra apenas um ponto.

Todas estas operações deverão ser efectuadas usando IOCTL, de acordo com os comandos especificados em 1.5 c).
Se o número de parâmetros não for o correcto, ou não for conhecido deverá ser impressa a mensagem:

```
Usar com:
dskm [users | sectors | secsize | version | geo | <LBA sec. no. to show ]
```

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp2.pdf

2 Entrega do trabalho

2.1 Data

A data limite de entrega é terça-feira 13 de Junho de 2006. A entrega para além do prazo sofre uma penalização de 0.5 valores por dia.

2.2 Código fonte

O ficheiro com o código fonte do módulo deverá ter o nome **dsk.c** e deve ser acompanhado de um *makefile* para gerar o módulo com o nome **dsk.ko**. O módulo deverá ser compilado de modo que com o comando **modinfo** obtenha-se informação pelo menos sobre a licença, a descrição dos parâmetros (indicando o valor por defeito), uma descrição breve do módulo, a versão, e os Nomes, N$^{os.}$ e curso dos autores:

```
modinfo dsk.ko

filename:       dsk.ko
license:        GPL
author:         Helder Daniel
version:        1.0
description:    Leitor do disco rígido IDE0
vermagic:       2.6.14-ipipe preempt 386 gcc-3.3
depends:
srcversion:     B865AF6B744DFDB7321CE85
```

De qualquer forma, todos os ficheiros devem ter como comentário os Nomes, N$^{os.}$ e curso dos elementos do grupo.
Deve ser entregue também o script **dsk** que insere e remove o módulo do kernel, referido em 1.4.

Pode ser entregue por e-mail ou em disquete ao docente do turno prático em que está inscrito o grupo.

2.3 Relatório

Com o código fonte terá de ser entregue um relatório, que deverá incluir um pequeno manual do utilizador e um manual de implementação. Um *template* para este relatório está disponível na página da disciplina:

http://w3.ualg.pt/~hdaniel/pin/avaliacao/relat_template.pdf

No manual do utilizador deverá ser descrito tanto o módulo **dsk.ko** como a aplicação **dskm**, referida no ponto 1.7, incluindo os parâmetros da linha de comandos de ambos.

O relatório deverá ser entregue em papel, ao docente do turno prático em que está inscrito o grupo.

3 Avaliação

De acordo com o indicado na aula teórica 1. Será tomada especial atenção à correcção do programa, à eficiência da implementação e à estrutura e clareza do código.

Nota: Funcionalidades não pedidas nos enunciados não serão avaliadas.

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp2.pdf

4 Bibliografia recomendada

Corbet, Jonathan, Alessandro Rubini e Kroah-Hartman, Greg (2005). *"Linux Device Drivers 3rd edition"*, O'Reilly, http://lwn.net/Kernel/LDD3/

Salzman, Peter Jay, Michael Burian and Ori Pomerantz (2005). *"The Linux Kernel Module Programming Guide"*, *http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf*

Identificação de dispositivos IDE: *http://www.t13.org/technical/e00159r2.pdf*

Aulas teóricas e práticas disponíveis em http://w3.ualg.pt/~hdaniel/Ed/, especialmente:

Capítulo 5 e 7 das aulas teóricas.
Ficha prática 4.2 e 6.

Nota:
podem ser utilizados ou adaptados funções e scripts desenvolvidas nas aulas teóricas, práticas ou dadas como código auxiliar para as fichas práticas, em particular poderá ser usado o código auxiliar da RAMDSK como base de desenvolvimento do módulo, mas todas as funções e timers que simulão a remoção do disco terão de ser removidas, e o nome das função deverá começar por DSK e não RAMDSK.